ES 4 Lab 6: Reading the buttons of an NES controller

Prelab due 24 hours before your lab session, 24 hours before your lab, April 11-15

1 Introduction

In this lab, you'll build the interface for an NES controller, which uses a simple 3-wire protocol to read the state (on or off) of 8 buttons. The output of your design will be 8 digital signals, which can drive some LEDs... or anything else you dream up.

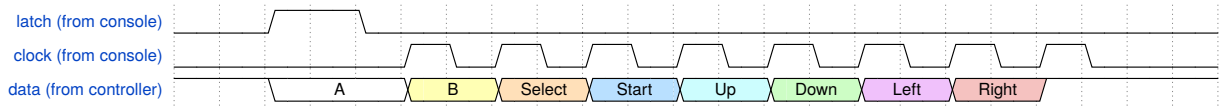
After successfully completing this lab, you should be able to:

- Generate a non-trivial output waveform using combinational and sequential logic
- Read data from a serial interface and convert it to a parallel output

2 Prelab

The internals of the NES controller are very simple: it's a shift register connected to 8 buttons. When a "latch" signal is asserted, the values of the buttons are stored in the shift register. Then, when the shift register is clocked, the values appear on the output one by one. This allows the console to read 8 independent buttons using only 3 wires (plus power and ground).

The basic waveform is shown below:



To read the buttons, we have to solve several problems:

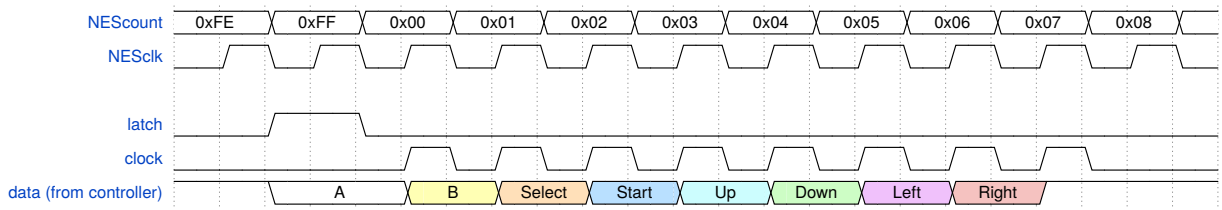
- Divide the FPGA clock (48 MHz) down to something the controller can handle. The NES console appears to use 83.3 kHz (clock period of 12 µs), but pretty much anything less than 3 MHz will work.
- Execute the following, somewhere between 10 and 100 times per second:
  - Drive the latch signal high for 1 clock period (although the clock signal is not being sent to the controller during this time).
  - Drive the clock signal for 8 cycles.
  - Read the data signal synchronized with the clock.

P1: Create a counter in VHDL driven by the 48 MHz HSOSC clock which rolls over about once every 25 ms (between 40 and 50 Hz). We will use this counter to orchestrate the button-reading process, and it will repeat every time the counter rolls over.

If we take bit 8 of the counter from P1, we get an oscillating signal that is suitable for the NES clock signal[1]. We'll call this `NESclk`. Likewise, if we take bits `N downto 9` of the counter, we get a slower (and smaller) counter which is just a count of the cycles of `NESclk`. Let's call this `NEScount`. We're going to ignore bits `7 downto 0` of this counter; they're changing too quickly to be useful.

These two derived signals are shown below for a small snippet of time, along with the signals you'll need to generate:

[1] Normally it's not good practice to do this on an FPGA, but it's not a problem because our design is so small.



P2: Write a line of VHDL code to generate the latch signal as a function of `NESclk` and/or `NEScount` (yes, you should be able do to this in one line). The figure above shows one particular alignment of the latch signal relative to `NEScount`; you can pick something else if it makes more sense to you.

P3: Write VHDL code to generate the controller clock signal as a function of `NESclk` and/or `NEScount` (again, you should be able do to this in one line).

If you have doubts about your code, you can always try it in Modelsim/Questasim or GHDL!

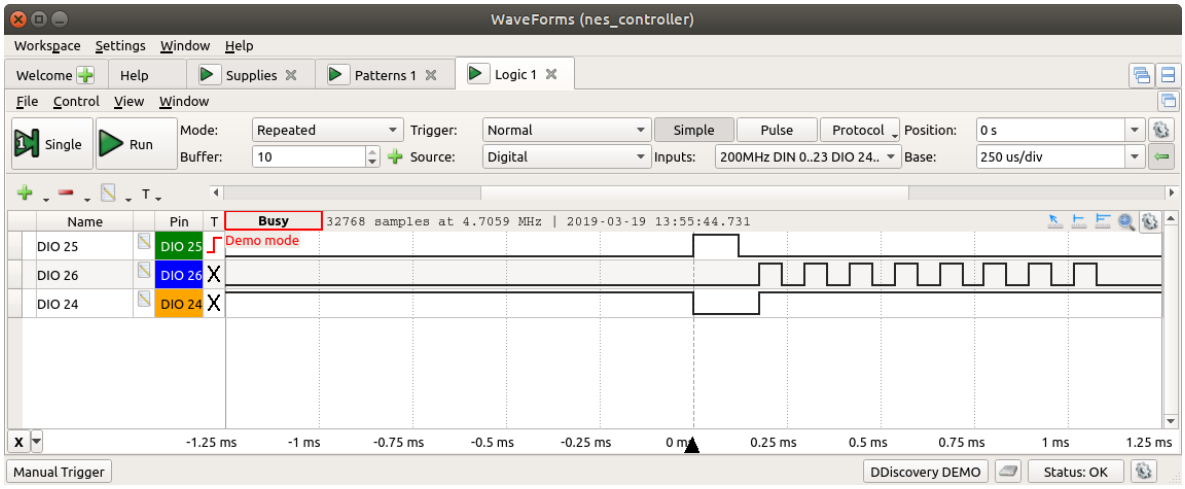
3 In lab

L1: Connect up the NES controller. We've already done the hard work of putting a breadboard-compatible connector on it, so just plug it in.

For the controllers we have, the signals are:

Power (3.3V)	Red
Ground	Yellow
Clock	Blue
Latch	Black
Data	Green

You may want to start by experimenting with the Digital Discovery to see what the waveforms should look like. A Waveforms signal-generator script that creates the correct output for the NES controller is posted on the course website. If you connect the controller to the Digital Discovery and run the signal generator and logic analyzer, you should be able to see the waveform change as you push the buttons. The pin mapping is 25→Latch, Clock→26, Data→24.



L2: Integrate the bits from your prelab to generate the output signals. Use the Digital Discovery to check that your output signals are actually generated correctly, and that the controller is responding.

L3: Write the code to grab the data signal in a shift register. When you've shifted in all 8 of the values, the result should get copied into another register. This is to prevent spurious values from affecting your output as you shift the register (e.g., if button A is pressed, it will shift through all the positions in the shift register — you don't want it to appear that the other buttons are pressed, however briefly.)

L4: Connect the output signals to something; perhaps LEDs or segments of your 7-segment display. In addition to taking a picture of the circuit, capture appropriate screenshots in Waveforms to demonstrate that your solution works.

4 What to turn in

Your lab report should contain:

- Standard "front matter" (see the lab reports handout).
- A photograph of your completed circuit.
- Your complete VHDL code.
- Your debugging log, including intermediate tests you performed (including ones that failed), steps you took to debug your design, and problems you encountered.
- Your plan(s) for testing your design, and any test results you recorded.
  - What was the most valuable thing you learned, and why?
  - What skills or concepts are you still struggling with? What will you do to learn or practice these?
  - How long did it take you to complete the lab? This will help calibrate the workload for future iterations of the course.